TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik | Institut für Software- und Multimediatechnik | Lehrstuhl für Softwaretechnologie

Managing Distributed Context Models Requires Adaptivity too

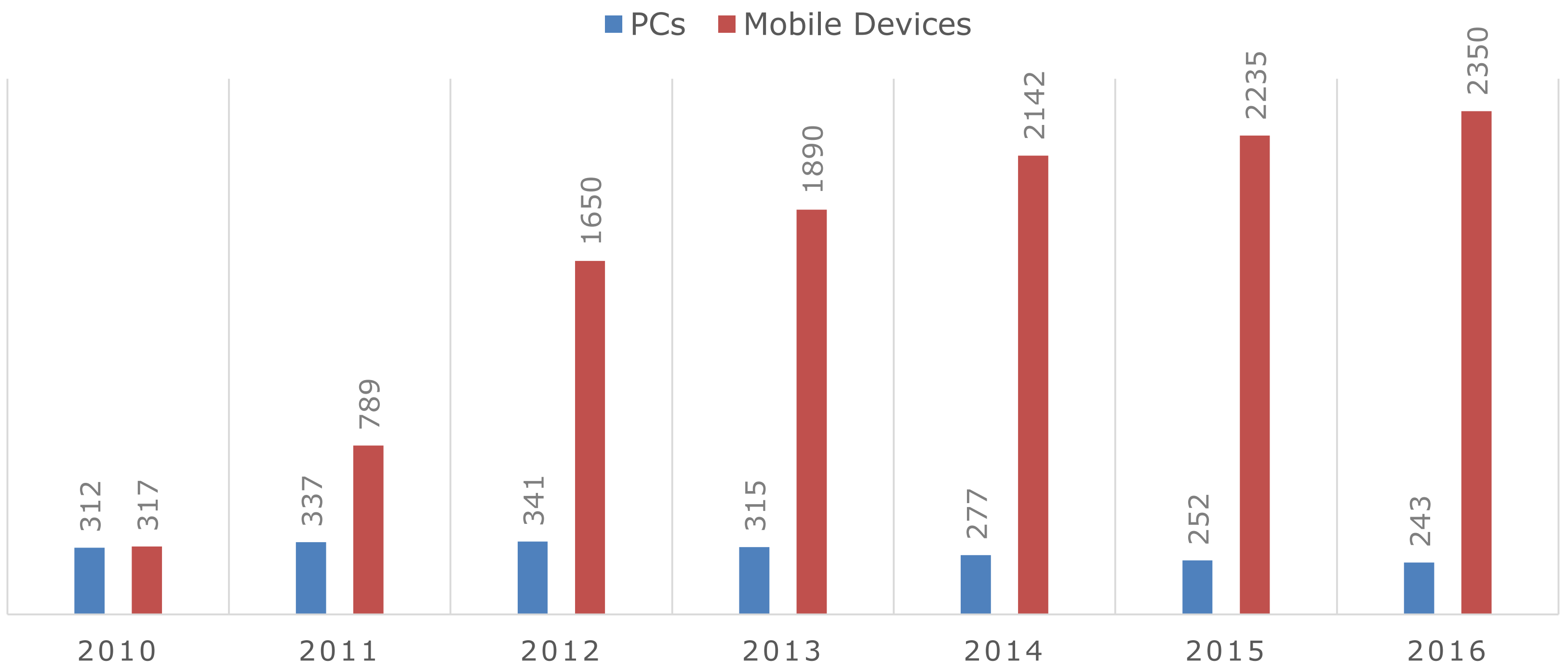
Technische Universität Dresden
Software Engineering Group

Christian Piechnick, Maria Piechnick, **Sebastian Götz**, Georg Püschel and Uwe Aßmann

DRESDEN concept Exzellenz aus Wissenschaft und Kultur

TECHNISCHE UNIVERSITÄT DRESDEN

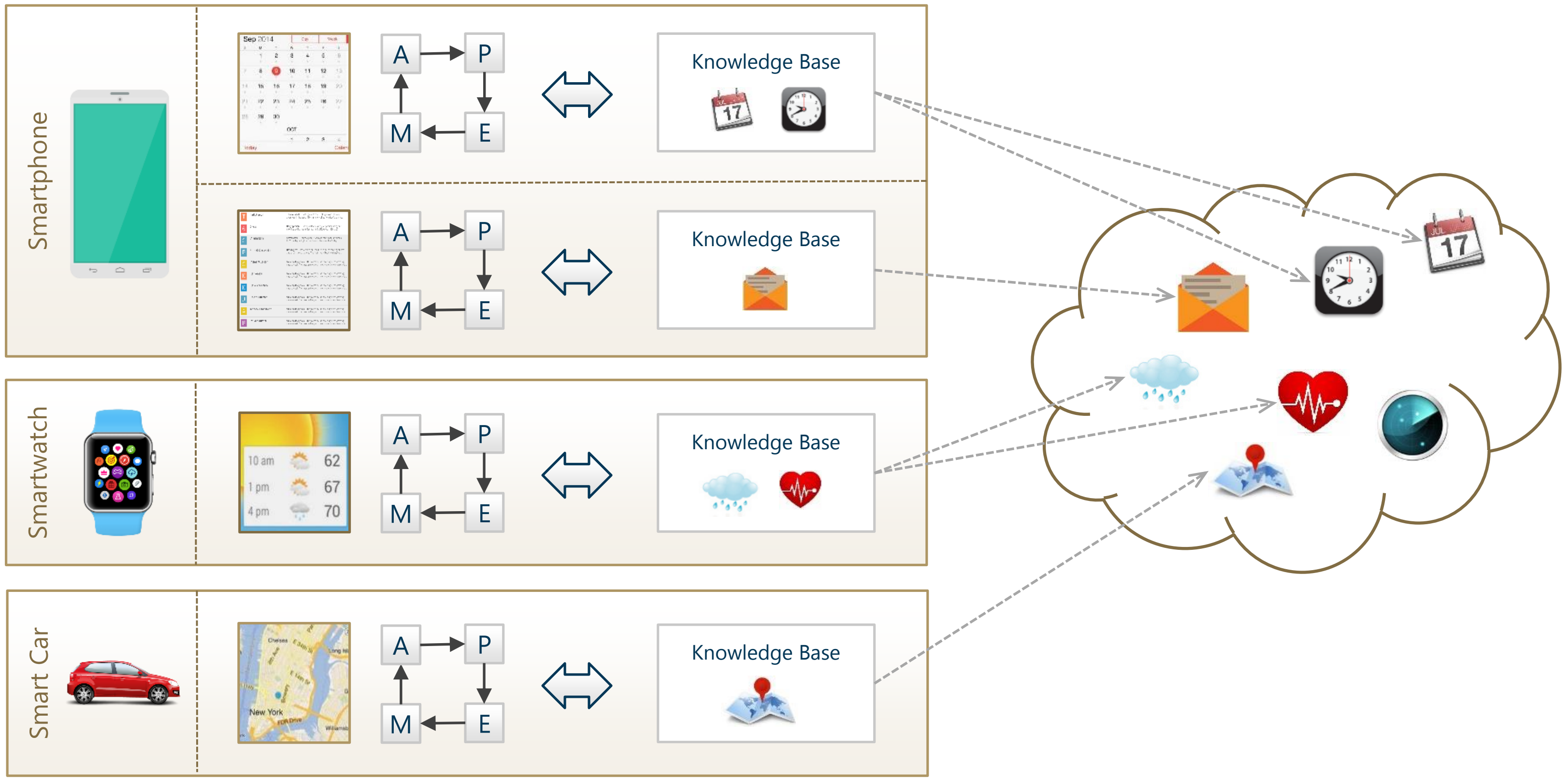
Trend Towards Mobile Computing Devices



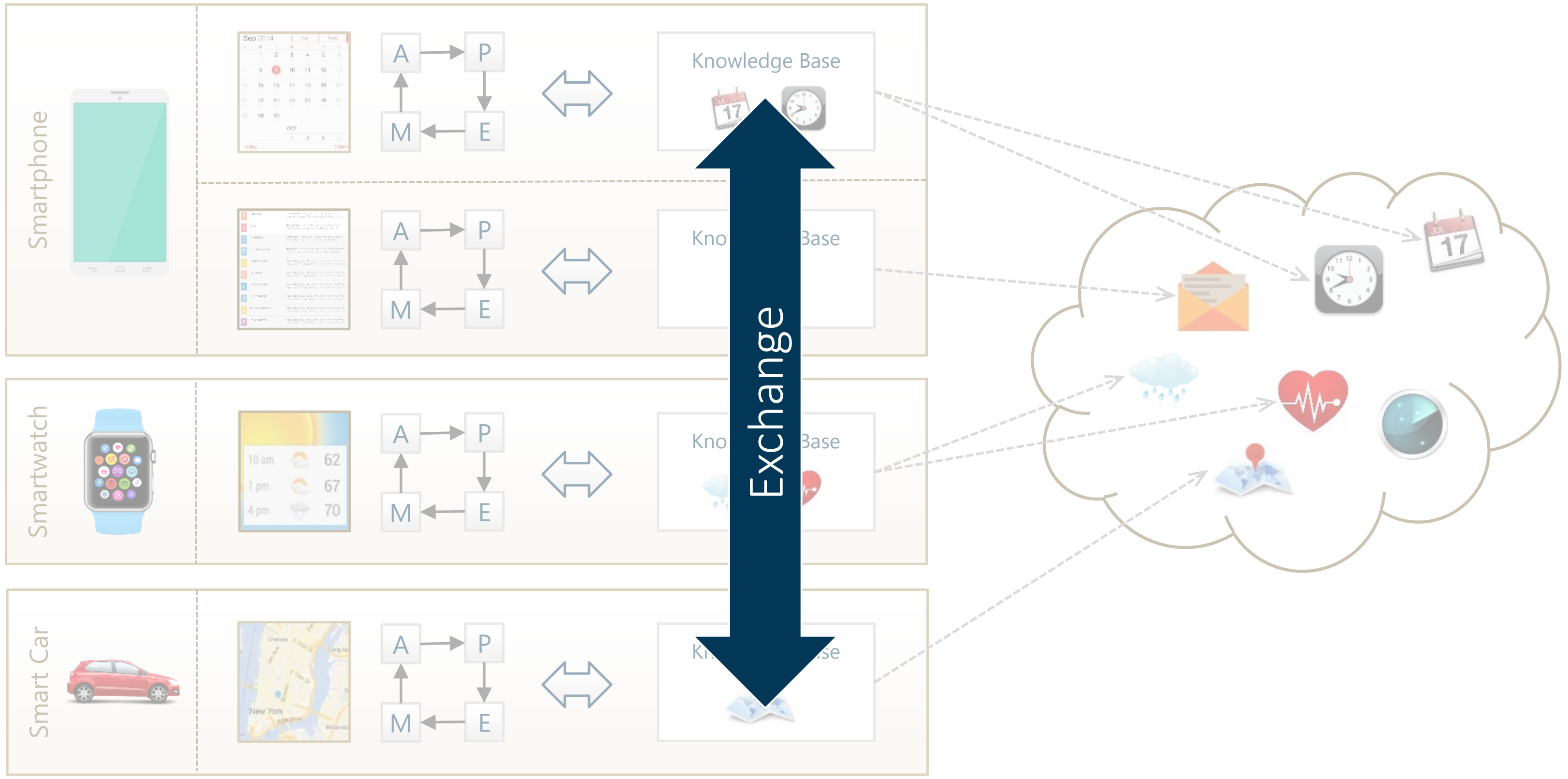
1000 Sales of devices (Source Gartner)

Mobility = Changing Contexts = Changing Requirements

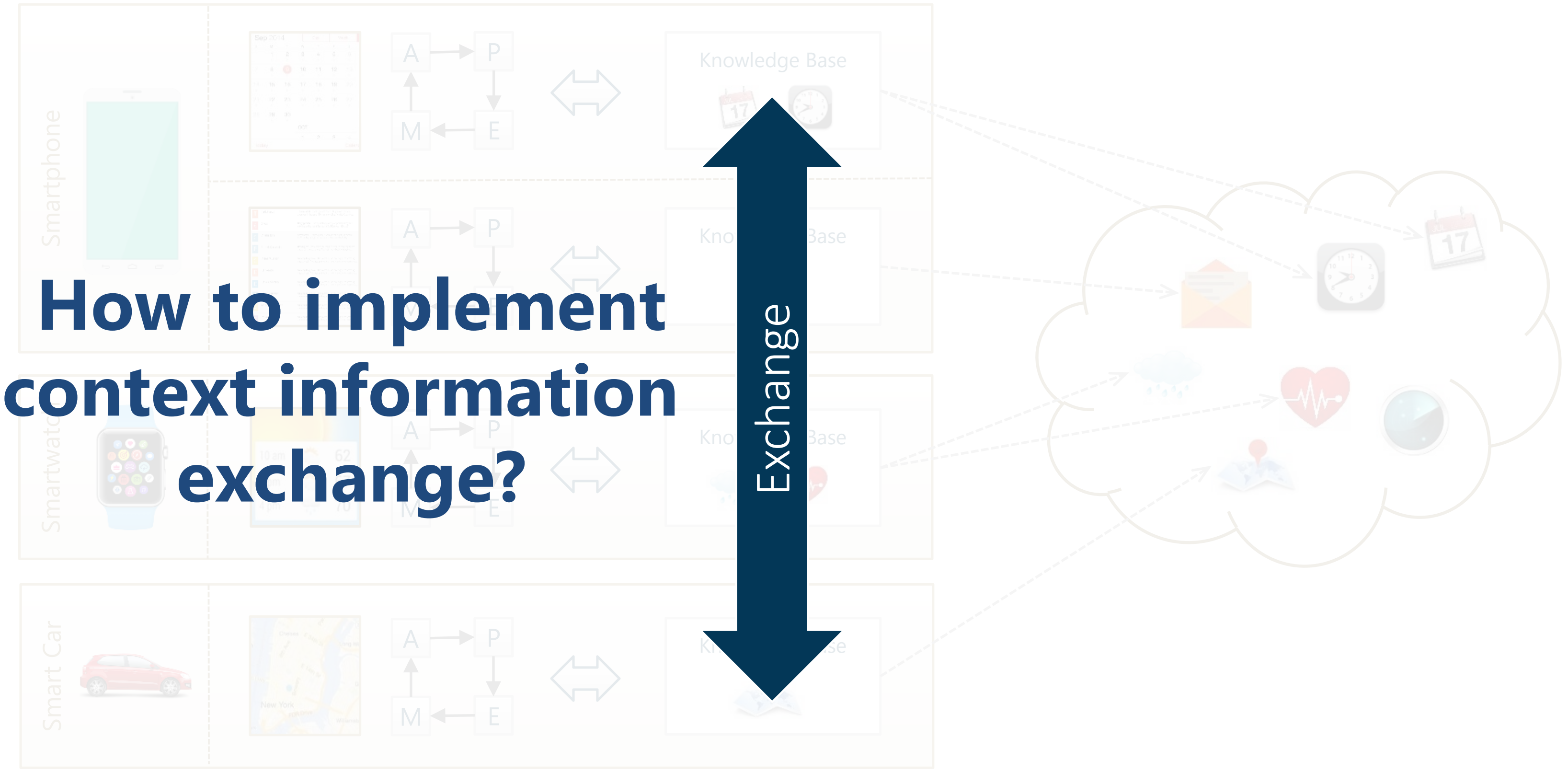
Individual apps may have different context information



Individual apps may have different context information



Individual apps may have different context information



How to implement context information exchange?

TECHNISCHE UNIVERSITÄT DRESDEN

Managing Distributed Context Models Requires Adaptivity too

VARIATION DIMENSIONS

Approach

Literature Study

- Investigation of 16 publications of technologies with the ability to exchange context information
- Result: Different strategies are used for different aspects
    1. **What** context information is accessible
    2. **When** context information should be exchanged
    3. **Who** initiates the exchange of context information
    4. **How** should context information be managed
    5. **Where** should context information be managed
- Concrete strategy depends on concrete requirements (may change at runtime)

  → *e.g., data traffic, expressiveness, size of the models, performance, ability to handle privacy constraints*

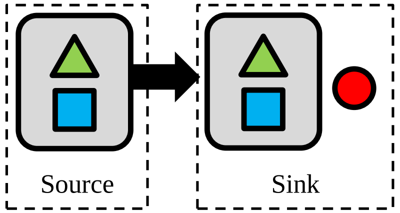
  → Context model management must itself be adaptive

  → Meta-Adaptation required

1) What context information is accessible

1.A Complete

- Most common solution
- Sink as full access to the context model of the source

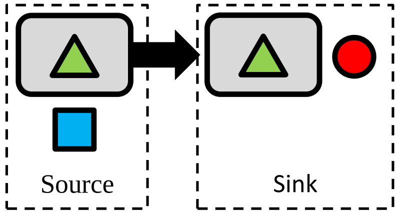


- Sink can decide which information is relevant
- Privacy issues cannot be addressed
- Potentially a large amount of data traffic

1) What context information is accessible

1.B Partial

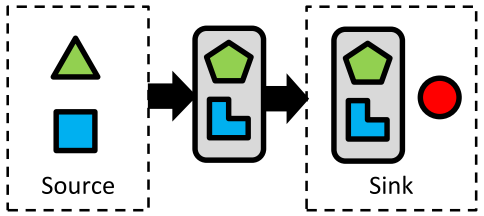
- Not all context information *should be accessible by* or *are relevant for the sink*
- Access to information might restricted
  → Privacy issues can be addressed
- Sink has the option to exclude irrelevant information
  → Data traffic might be reduced
- Higher complexity



1) What context information is accessible

1.C View-Based

- Source provides views on the context model
- Explicit handling of privacy issues
- Reduction of data traffic due to potential abstraction
- Views can be defined by the source or sink

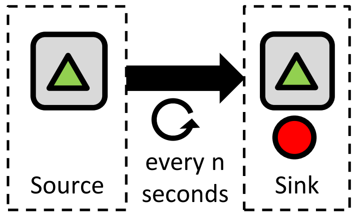


2) When context information should be exchanged

2.A Periodically

- Information is pulled or pushed in certain time intervals
- Update frequency defined statically or dynamically

- Easy to implement
- Potentially unnecessary data traffic due to transfer of unchanged information
- Subsampling must be prevented

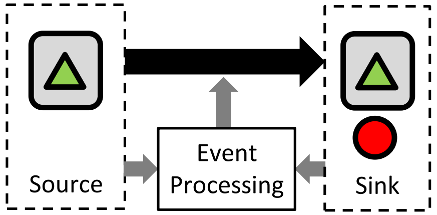


2) When context information should be exchanged

2.B Event-Based

- Source and/or sink can produce events
- Event processing leads to context information exchange

  e.g., a certain value changed a certain amount



- Reduce data traffic
- Prevent subsampling
- Introduction of further complexity

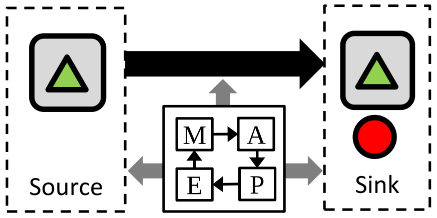
2) When context information should be exchanged

2.C Context-Based

- Context-dependent exchange
- Feedback loop decides based on context information

  e.g., two devices are very close

- Higher complexity
- Higher flexibility (auto-tune data-traffic etc.)



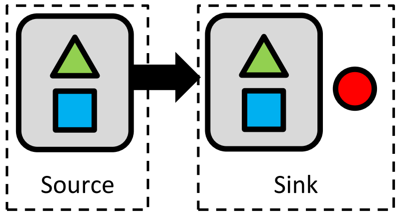
3) Who initiates the exchange of context information

3.A Source

- Source proactively distributes context information

- Source has full control what data is distributed
  → improves privacy issue handling
  → may reduce data traffic (e.g., only pushed when value changed)

- Sink may not be able to specify what information is relevant
  → may increase necessary data traffic



3) Who initiates the exchange of context information

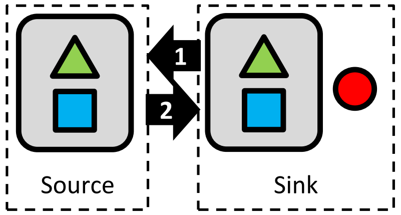
3.B Sink

- Sink pulls context information
- Source sends information as response

- Source has full control what data is distributed

  → improves privacy issue handling

  → may reduce data traffic (e.g., only pushed when value changed)
- Sink may not be able to specify what information is relevant

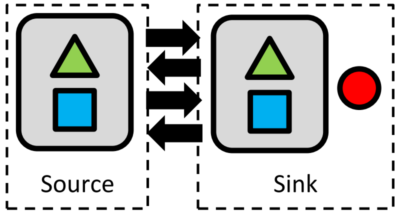


3) Who initiates the exchange of context information

3.C Negotiation

- Combination of source- and sink-based initiation in a black-board architecture
- Sinks can access the context model via a query interface
- Source might grant or deny access and might offer views
- Sinks may be able to register for certain events and get notified

- Higher complexity
- Higher flexibility

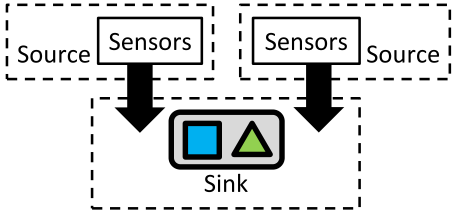


TECHNISCHE UNIVERSITÄT DRESDEN

4) Where should context information be managed

4.A Centralized

- Central server manages one context model for multiple clients
- Every updated value is sent to the sink



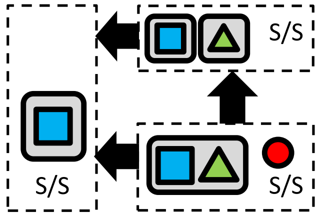
- Reduction of the number of required connections
- Reduction of data traffic
- For devices with limited resources (e.g., main memory) this might be beneficial

- Single point of failure
- Potentially large single model
- Handling privacy issues gets complicated

4) Where should context information be managed

4.B Decentralized

- Applications manage their own context model and are able to exchange context information with other applications



- Handling privacy issues possible
- Decrease the size of the individual models (compared to the centralized approach)

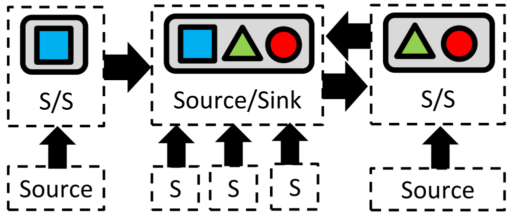
- Higher number of required connections
- Potentially decreased data traffic

4) Where should context information be managed

4.C Hybrid

- Combination of centralized and decentralized approaches
- Multiple central sinks that are connected in a peer-to-peer network
- Applications can be grouped in a centralized style while the sinks can exchange context information
- Concrete architecture might be defined statically or organized dynamically

- Possible to dynamically combine the benefits of both approaches

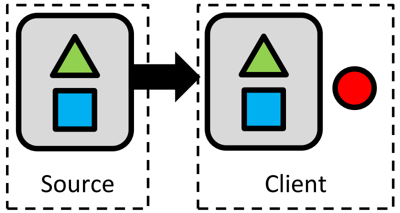


5) How should context information be managed

5.A Copy

- Sink copies the received information into the own context model
- Most common strategy

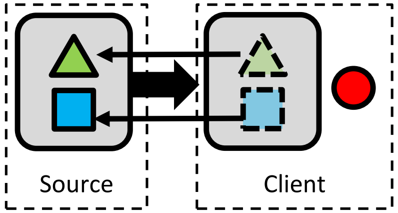
- Easy to implement
- Suitable when number of reads exceed to the number of writes



5) How should context information be managed

5.B Proxy

- Sink stores a reference to the actual (remotely available) information
- Every read gets transformed into a remote call



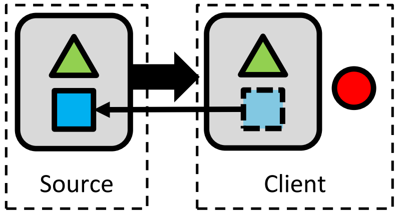
- Size of the stored data is decreased
- Decrease data traffic when the number writes exceed to the number of reads

- Evaluation performance is decreased

5) How should context information be managed

5.C Hybrid

- Some information is copied other managed by proxies
- Dynamic decision based on read/write characteristics



- Optimization of data traffic
- Optimization of the size of the managed models

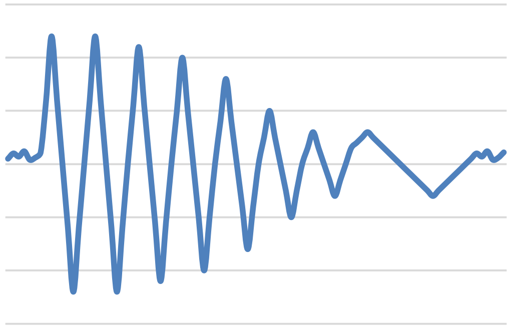
- Higher complexity (additional monitoring components required)

GOAL: Show feasability

- Only a small first example

- **Domain**: Blended Interactive Spaces (Multi-Device Interaction)
- **Use Case**: *Bump-to-Give Interaction Pattern*
  When two devices are bumped together, content (e.g., an image) is transferred from one device to the other

- **Recognition**: A special function on accelerometer data
  → Both devices must show this pattern at the same point of time

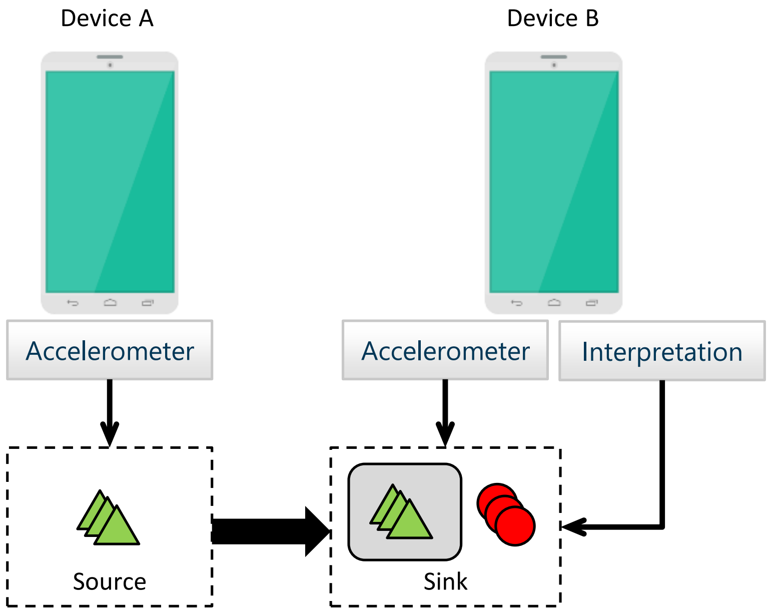


Copy-based

- Interpretation of the data of both devices on *Device B*
- Copying the accelerometer data from *Device A* to *Device B*

- Very easy to implement
  → Synchronization etc. can be done on one device

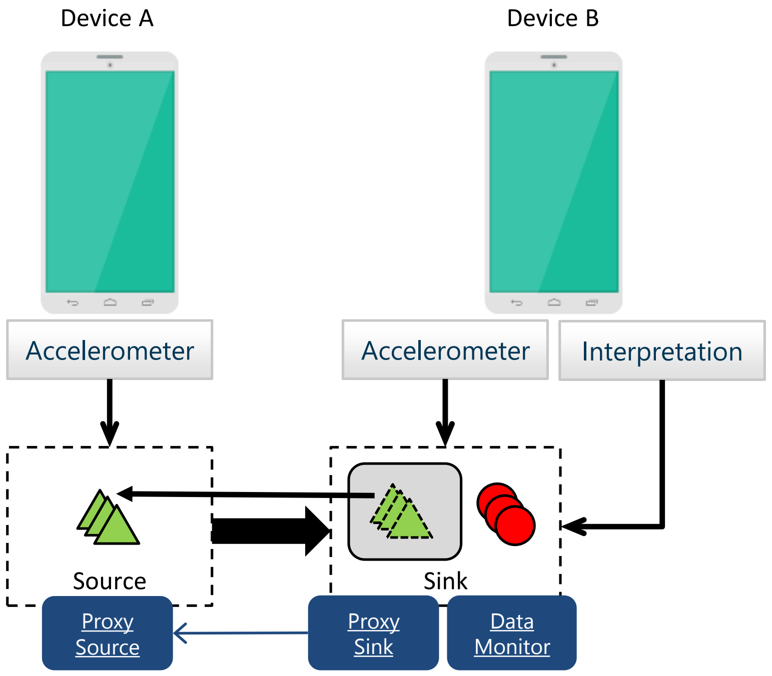
- **Data Traffic: 20kB/s**
- Data of *Device A* is only read when the interpretation of the data of *Device B* detects a bump



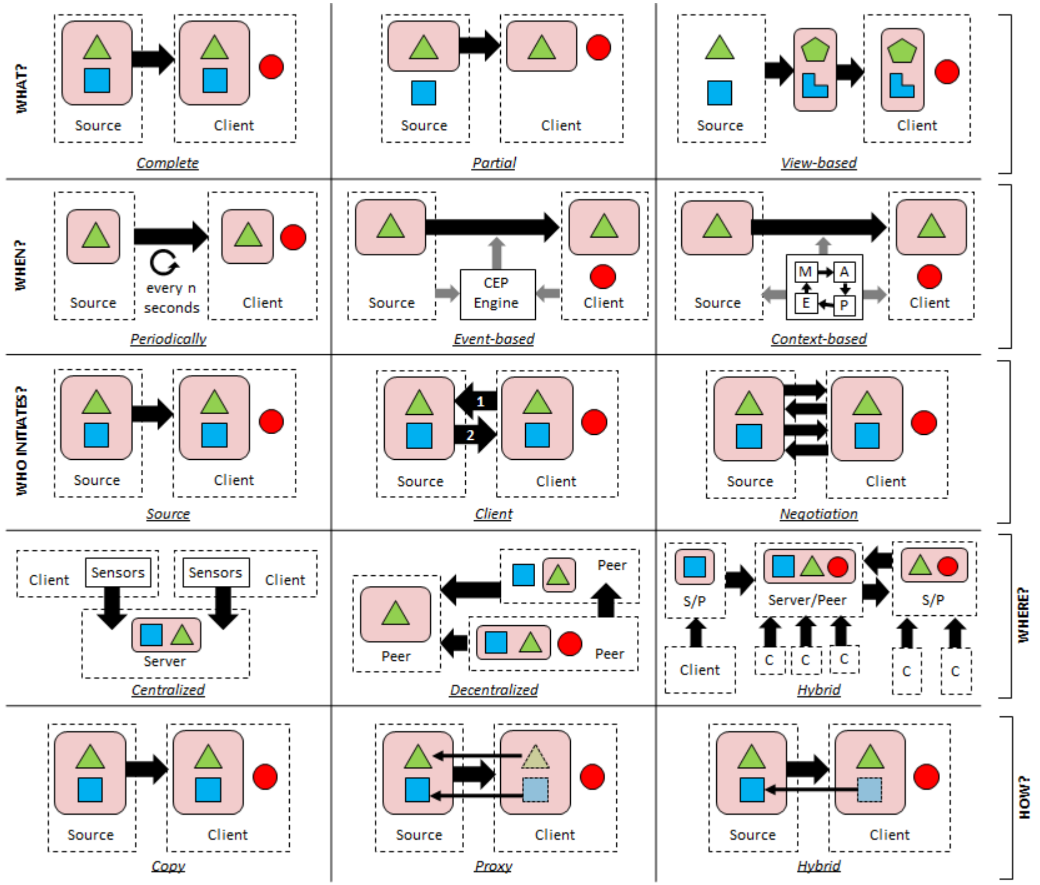
Proxy-Based

- DataMonitor roles monitors read/write access of context information
- When read/write actions exceeds a ration of 1:2, values are no more copied but a reference stored
  → Actual values are no longer transferred
  → When information is read, a remote call is generated

- **Data Traffic: Almost 0kB/s**
- Interpretation takes a little bit longer (337ms more)
  → hardly recognizable



Thank You!


WHAT?
Source
Client
Complete
Source
Client
Partial
Source
Client
View-based
WHEN?
Source
every n seconds
Client
Periodically
Source
CEP Engine
Client
Event-based
Source
M
A
E
P
Client
Context-based
WHO INITIATES?
Source
Client
Source
1
2
Source
Client
Client
Source
Client
Negotiation
Client
Sensors
Sensors
Client
Server
Centralized
Peer
Peer
Peer
Decentralized
S/P
Server/Peer
S/P
Client
C
C
C
C
C
Hybrid
WHERE?
Source
Client
Copy
Source
Client
Proxy
Source
Client
Hybrid
HOW?